Python for Data processing

# Lecture 2:
# **Arrays, tensors and computations - Part II**

Glib Ivashkevych

# What we already know

**A lot about numpy arrays:**

- creation
- indexing
- universal functions
- linear algebra
- best practices
- I/O

# Why numpy is not enough

**numpy arrays are great but:**

- they work only on CPU
- they provide only basic building blocks

**For deep learning:**

- CPU/GPU/TPU/?
- gradients

# PyTorch

- **tensors** provide the same operations as `numpy` arrays
- work on CPU/**GPU**/**TPU**
- provide **autogradients**
- **deep learning** building blocks
- efficient **data loading**
- **deployment**

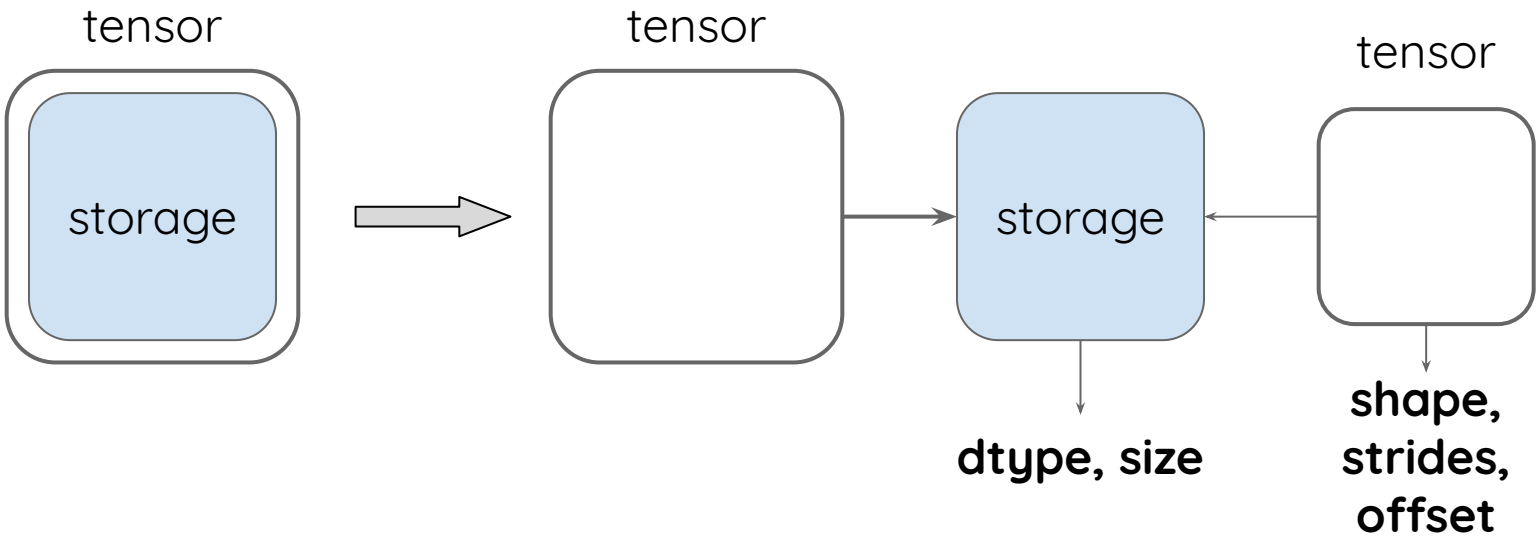
# PyTorch tensors

# Tensors

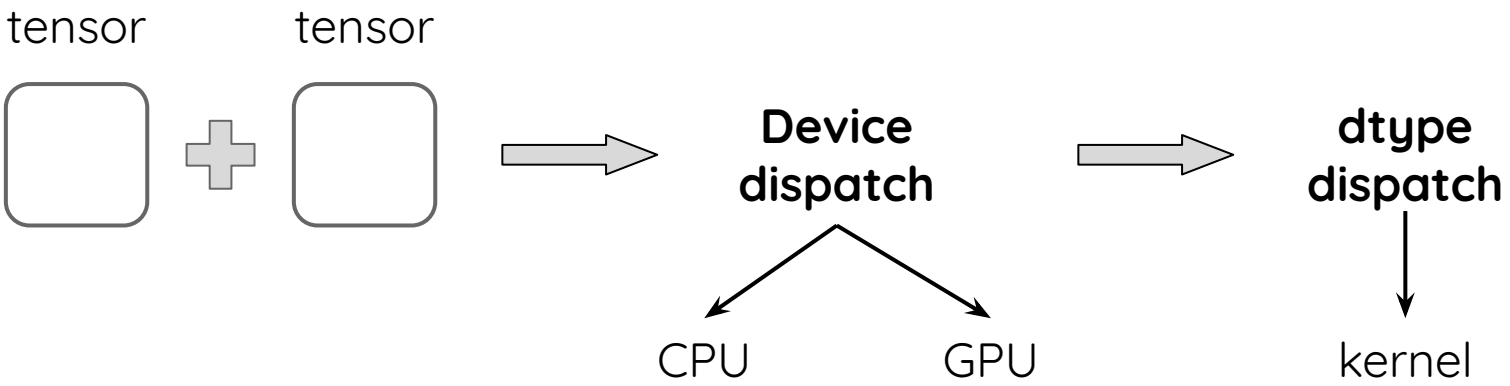
- similar to arrays, provide the same computational facilities
- can **share** data
- can live on **different devices**
- provide **declarative** computations

→let’s try it out!

# Tensors and storage



# Devices and computations




→let's try it out!

# View, copies, reshaping

PyTorch is a bit more elaborated:

- `view`: always returns a view or fails
- `reshape`: returns either view or new tensor
- depends on contiguity constraints

→let's try it out!

# Gradients

In deep learning we need **gradients**:

- to calculate updates to network parameters (weights)
- no way to do that in NumPy
- an easy go in PyTorch (a bit more elaborated in Tensorflow)

$$L\left(a_{ij}\right) \text{ (scalar)} \rightarrow \frac{\partial L}{\partial a_{ij}} \text{ (tensor)}$$

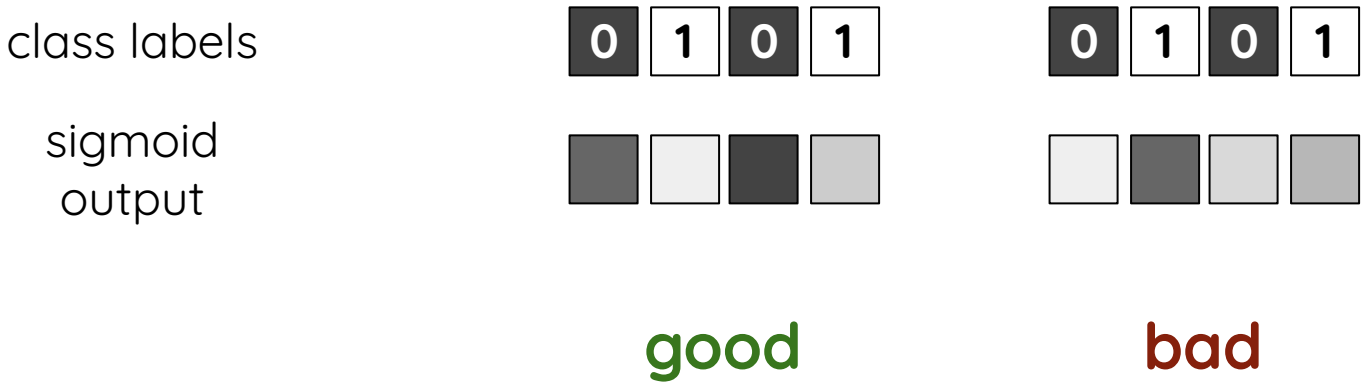
→let's try it out!

# Logistic regression with PyTorch

# Logistic regression: setup

- **two-dimensional** input (created with `make_blobs` from `sklearn.datasets`)
- **binary** classification with **linear** decision boundary
- output: **sigmoid**
- from **scratch**

# Log loss




**We want: probability** ↓ for class 0, probability ↑ for class 1

→let's try it out!

# What we've learned

- **numpy** broadcasting
- how to use **numpy** efficiently
- PyTorch tensors and gradients
- how to perform simple gradient descent

# Words we know

- **Jupyter**
- numpy
- PyTorch

# Assignment

- explore NumPy broadcasting and linear algebra
- reimplement logistic regression with PyTorch deep learning building blocks

# questions?